UNLIMITED

(2)

RSRE
MEMORANDUM No. 4467

ROYAL SIGNALS & RADAR ESTABLISHMENT

SELF-SUPERVISION IN MULTILAYER ADAPTIVE NETWORKS

Author: S P Luttrell

PROCUREMENT EXECUTIVE,
MINISTRY OF DEFENCE,
RSRE MALVERN,
WORCS.

RSRE MEMORANDUM No. 4467

UNLIMITED

CONDITIONS OF RELEASE

0118010 308061

..... DFNC U

..... DFNC Y

Reports quoted are not necessarily available to members of the public or to commercial organisations.

Defence Research Agency, Electronics Division

Royal Signals and Radar Establishment

Memorandum 4467

Self-Supervision in Multilayer Adaptive Networks

Stephen P Luttrell

Image Processing Research Section
Defence Research Agency
Royal Signals and Radar Establishment
St Andrews Rd, Malvern, WORCS, WR14 3PS
Janet: luttrell@uk.mod.hermes
Internet:luttrell%hermes.mod.uk@relay.mod.uk

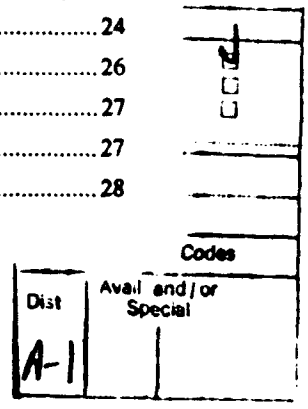
6 December 1991

Abstract

We theoretically derive and numerically simulate a new phenomenon called *self-supervision*, in which the higher layers of a multilayer *unsupervised* network control the optimisation of the lower layers, even when there is *no* external supervising teacher present. Self-supervision is a very convenient hybrid, which combines the best properties of unsupervised and supervised network training algorithms.

THIS PAGE IS INTENTIONALLY LEFT BLANK

Contents



List of figures

1. Introduction

A common criticism of unsupervised adaptive networks is their poor performance as classifier networks (i.e. supervised networks). In [1] the so-called "learning vector quantisation" (LVQ) method was introduced in which an external teacher supervised the output of a vector quantiser (VQ), and steered it towards a desired target output. This is a hybrid approach, using both unsupervised and supervised elements in its training algorithm, and it has met with limited success because the ability of a VQ to tailor sophisticated class boundaries is rather limited. Basically, a VQ is a single stage network with very limited capabilities; we must use a multistage VQ to enhance the performance[1].

The novel result that we present in this memorandum is that it is not necessary to introduce an external teacher in order to introduce supervision into an unsupervised network. It turns out that multilayer adaptive networks can supervise themselves *even though* they are trained overall as unsupervised networks. For simplicity, we study the theory of multistage VQ networks that we developed in [2, 3, 4, 5], in which the higher layers supply feedback signals to assist in the optimisation of the lower layers, although there is no external teacher present[2].

Our network is well suited to the problem of low-level image processing, where information at each length scale should be processed in the light of contextual information at longer length scales. There are many ways of implementing contextual processing, but our multistage approach to this problem distinguishes itself by scaling well to high-dimensional problems[3].

In [6, 7] we successfully apply our theoretical results to time series and image compression, respectively, and in [8, 9] we solve the problem of detecting statistically anomalous features in statistically homogeneous backgrounds. Self-supervision should improve the performance of the network in these applications, because it extends the sequential layer-by-layer optimisation that we used in [6, 7, 8, 9] to a full global optimisation of the multilayer network[4]. Furthermore, the "top-down" information pathways that self-supervision uses are the same as those required by an LVQ-like supervised network, which allows us easily to extend our approach to become a full classifier network.

In Section 7 we present a simple diagrammatic review of vector quantisation, and its extension to 2-stage vector quantisation. In Section 3 we perform some numerical simulations to demonstrate the self-supervision effects that emerge in a 2-stage VQ.

In the appendix we review the general subject of VQ's, and we derive the properties of 2-stage VQ's in the limit of a large codebook size (i.e. the continuum limit). This derivation is rather technical, but it is the central theoretical result upon which self-supervision depends.

---

[1] This argument is analogous to the perceptron versus multilayer perceptron argument.

[2] The theory can be extended to an LVQ-like hybrid in which an external teacher is present. However, we find it very appealing to think of the external teacher as merely the influence of those layer(s) of the adaptive network that lie beyond the last one that we explicitly simulate.

[3] The network architecture superficially resembles the architecture of the primate visual cortex. This is not accidental! In the future we propose to develop these ideas into an architecture that looks much more like a visual cortex.

[4] The problem of local minima is not solved by self-supervision.

2. Vector quantisation model

In the appendix we collect together some background theory on VQ's. In this section we present a diagrammatic summary of this theory.

2.1. Diagrammatic interpretation of vector quantisation

In a VQ the compression and reconstruction process may be interpreted in terms of encoding and decoding during transmission of information through a noiseless communication channel[5], as shown in Figure 1.

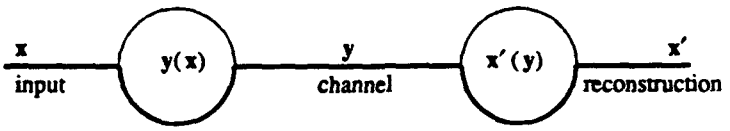


Figure 1. Communication channel encoding/decoding interpretation of vector quantisation.

There are two distinct methods of encoding that we should consider:

1. Nearest Neighbour Encoding (NN): Select the codevector that lies closest to the input vector, in the Euclidean sense.
2. Minimum Distortion Encoding[6] (MD): Model the noise on the communication channel, then select the codevector that on average produces the closest estimate of the input vector, in the Eucliden sense.

We can use NN encoding as an approximation to MD encoding when the signal-to-noise ratio on the channel is large[7]; this approximation is exact in the limit of vanishing channel noise.

We may generalise the simple encoder/decoder system to a set of nested VQ's. For instance, in Figure 2 we show a two-stage VQ.

In Figure 2 we transform the components of the input pair $\mathbf{x}=(x_1,x_2)$ to yield the corresponding components of the pair $\mathbf{y}=(y_1,y_2)$, which is we then input to the nested VQ, whose output components $\mathbf{y}'=(y'_1,y'_2)$ we use to reconstruct the corresponding components of $\mathbf{x}'=(x'_1,x'_2)$. In [3, 4, 5] we present an approximate method of training the channels $x_1 \rightarrow y_1 \cdots y_1' \rightarrow x_1'$ and $x_2 \rightarrow y_2 \cdots y_2' \rightarrow x_2'$ *independently*, by modelling them as a pair of noisy channel VQ's (as in Equation 10). Although in this scheme we

---

[5]Our results could indeed be applied to the optimisation of VQs for communication of information through noisy communication channels, but that is *not* the purpose of our research programme. Our primary motivation for using a VQ model is to obtain simple closed-form analytic solutions, which we may then use to develop our understanding of more complicated models in the future.

[6]In Luttrell [12] we derive the asymptotic density of code vectors of topographic mappings trained using the minimum distortion prescription of Equation 3, and we find that it is independent of the neighbourhood function, assuming *scalar* quantisation and mild monotonicity constraints on the neighbourhood function. In Luttrell [13] we present an informal derivation of this result for the *vector* quantisation case. When we let the width of the neighbourhood function decrease to zero we recover a standard VQ, which causes some of the asymptotic properties of VQs to be the *same* as those of topographic mappings, provided that we use the MD encoding rather than the NN encoding.

[7]Caveat!. In the literature it is conventional to *always* use NN encoding, even though MD encoding is the correct procedure.

assume (incorrectly) that the channels do not mutually interfere, we nevertheless obtain useful results[8]

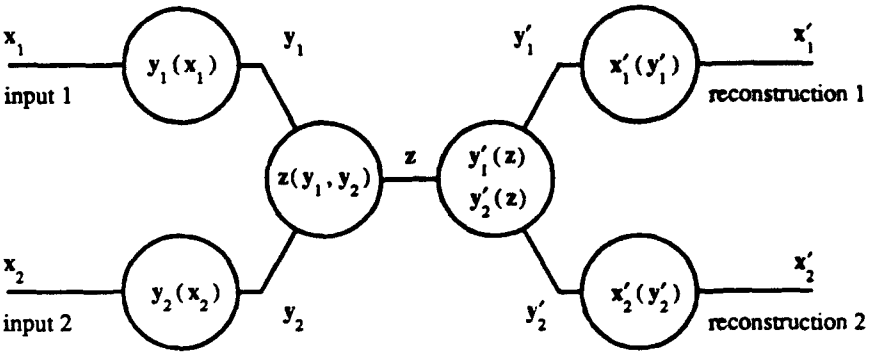


Figure 2. Two-stage vector quantisation. Two channels $x_1 \rightarrow y_1 \cdots y_1' \rightarrow x_1'$ and $x_2 \rightarrow y_2 \cdots y_2' \rightarrow x_2'$ are coupled through a common channel $(y_1,y_2) \rightarrow z \rightarrow (y_1',y_2')$.

We may further generalise the system in Figure 2 by creating a multi-stage structure of nested VQ's, which we may use for time series and image compression (see [6, 7]).

2.2. Ensemble average vector quantisation

We now consider the effect of mutual channel coupling on the optimisation of a nested VQ.

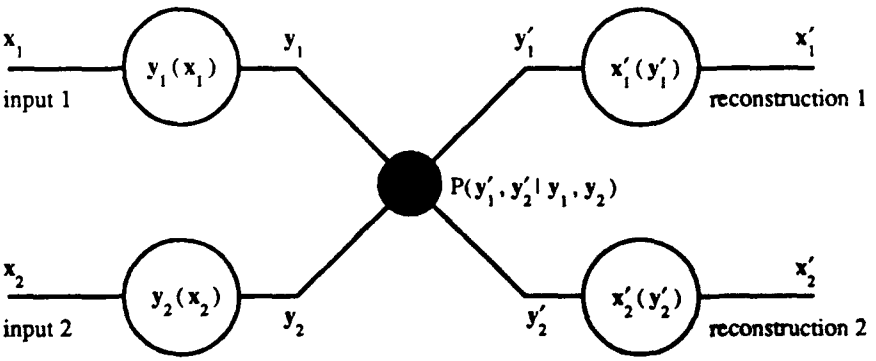


Figure 3. Ensemble average two-stage vector quantisation. $P(y'_1,y'_2|y_1,y_2)$ models the distortion due to the ensemble average of feasible $(y_1,y_2) \rightarrow z \rightarrow (y_1',y_2')$.

---

[8]For instance the very successful "anomaly detector" that we reported in [8,9] relies entirely on the independent channel assumption. In fact, it would have been impossible to obtain the very rapid training times (of order of 2 seconds per layer on a VAXstation 3100) without the independence assumption.

In Figure 3 we show an ensemble average version of Figure 2, where we take the average over realisations of the nested VQ $(y_1,y_2) \rightarrow z \rightarrow (y'_1,y'_2)$

The ensemble average approach is appropriate in situations where we continuously update the nested transformation $(y_1,y_2) \rightarrow z \rightarrow (y'_1,y'_2)$ as part of a training schedule. Ideally, we should optimise the $x_1 \rightarrow y_1 \quad y_1' \rightarrow x_1'$ and $x_2 \rightarrow y_2 \quad y_2' \rightarrow x_2'$ transformations to adapt to the changes in the $(y_1,y_2) \rightarrow z \rightarrow (y'_1,y'_2)$ transformation, but this is time consuming. Rather, it is better for us to arrange these transformations to adapt to the properties of the *ensemble* of $(y_1,y_2) \rightarrow z \rightarrow (y'_1,y'_2)$ transformations that might occur.

Using the marginal PDF's $P(y'_1|y_1,y_2)$ and $P(y'_2|y_1,y_2)$ of $P(y'_1,y'_2|y_1,y_2)$, we may write the expression for the distortion in Figure 3 as

$$D = \int dx_1\, dx_2\; P(x_1,x_2) \int dy'_1\; P(y'_1|y_1(x_1),y_2(x_2)) \|x'_1(y'_1) - x_1\|^2 \quad + \quad (1 \leftrightarrow 2) \quad (1)$$

We may interpret the various contributions to the expression for D by working from the outside of the expression to the inside as follows

1. The $\int dx_1\, dx_2\; P(x_1,x_2)\,(\ldots)$ integration averages over all the pairs of inputs (x_1,x_2) to channels 1 and 2, and $P(x_1,x_2)$ specifies the probability density with which each pair occurs.

2. The $\int dy'_1\; P(y'_1|y_1,y_2)\,(\ldots)$ integration averages over all the possible distortions of channel 1, due to the confluence of channel 1 and channel 2 in the nested VQ

3. $\|x'_1(y'_1)-x_1\|^2$ is the Euclidean distance between the input vector x_1 and its reconstruction $x'_1(y'_1)$ from the distorted version of channel 1

4. $(1 \leftrightarrow 2)$ denotes an analogous term for channel 2

In Figure 4 we represent diagrammatically in (y_1,y_2)-space (and (y'_1,y'_2)-space) the various terms of Equation 1.

We represent the contours of a typical $P(y_1,y_2)$, a typical $P(y'_1,y'_2|y_1,y_2)$, and the profiles of its two marginals $P(y'_1|y_1,y_2)$ and $P(y'_2|y_1,y_2)$. These marginals have a shape that depends on (x_1,x_2), which therefore mutually couples the contributions to the distortion in Equation 1, arising from the two transformations $x_1 \rightarrow y_1 \cdots y_1' \rightarrow x_1'$ and $x_2 \rightarrow y_2 \cdots y_2' \rightarrow x_2'$. It is both pleasing and economical that the ensemble average nested VQ in Figure 3 automatically determines the topographic neighbourhood functions for its $x_1 \rightarrow y_1 \cdots y_1' \rightarrow x_1'$ and $x_2 \rightarrow y_2 \cdots y_2' \rightarrow x_2'$ transformations, thus eliminating the need to introduce them by hand[9].

In the appendix we show how to minimise D by using a minimum distortion prescription in which we *simultaneously* optimise $y_1(x_1)$ and $y_2(x_2)$. We sometimes approximate this by using a nearest neighbour prescription.

---

[9]With hindsight, this is a powerful argument for using a simple VQ model in the first place. Had we attempted to use a more sophisticated type of model, we most likely would have missed this elegant result. Now we are alert to the possibility of topographic interpretations of more complicated models, where before we were ignorant of this possibility

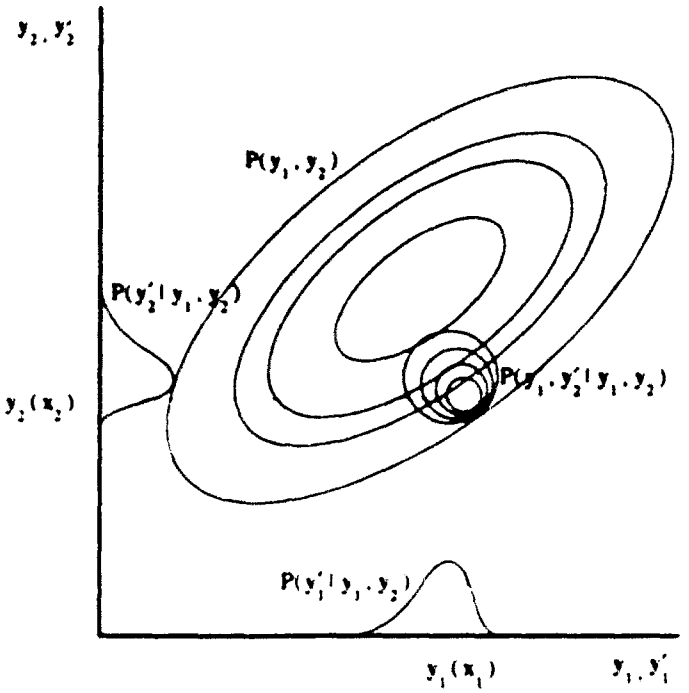


Figure 4. The marginal PDFs $P(y'_1|y_1,y_2)$ and $P(y'_2|y_1,y_2)$ of the ensemble average distortion $P(y'_1,y'_2|y_1,y_2)$ determine the topographic neighbourhood functions for optimising the $x_1 \rightarrow y_1 \cdots y_1' \rightarrow x_1'$ and $x_2 \rightarrow y_2 \cdots y_2' \rightarrow x_2'$ channels.

3. Numerical experiments

In this section we present a simple numerical simulation which demonstrates some of the benefits of self-supervision

3.1. Basic network operation

We run all of our numerical simulations using the network structure in Figure 3, with a 4-dimensional input vector $\mathbf{x}=(x_1,x_2)=(x_{11},x_{12},x_{21},x_{22})$, and with scalar outputs from the encoders $y_1(x_1)$ and $y_2(x_2)$. This is the minimal network that functions as a self-supervised VQ. In realistic applications we would expect much more complicated networks to be used, but they would all operate according to the principles demonstrated by the network in Figure 3.

In Table 1 we tabulate the various modes of operation that we use in the numerical simulation the we outline in Figure 5

	Nearest Neighbour Encoding	Minimum Distortion Encoding
Independent Channels	NN/I	MD/I
Correlated Channels	NN/C	MD/C

Table 1. Encoding and channel modes used in our numerical simulations. NN=nearest neighbour encoding, MD=minimum distortion encoding, I=independent channels, C=correlated channels.

In encoding mode NN we ignore the fact that $P(y'_2|y_1,y_2)$ and $P(y'_1|y_1,y_2)$ affect the resulting (y_1,y_2) in Equation 18, whereas in encoding mode MD we take full account of their influence. Note that the part of Figure 5 that is enclosed in a dashed box is the inner loop that handles the minimum distortion aspect of encoding mode MD. However, when we use encoding mode NN, we must still invoke step 3 (i.e. "compute distortion PDF's") of the simulation, because the distortion PDF's are required by step 6.

The two channel modes I and C test the effect of switching self-supervision off and on, respectively.

We now describe in greater detail each of the numbered boxes in Figure 5.

1. Clamp Layer 0 Inputs

Generate $\mathbf{x}=(x_1,x_2)=(x_{11},x_{12},x_{21},x_{22})$ using an appropriate random vector generating routine. We choose (x_{11},x_{12}) as a uniformly distributed random vector in a disc-shaped region, and then generate (x_{21},x_{22}) by rotating (x_{11},x_{12}) about the disc's centre by an random angle uniformly sampled from the interval $[-\theta,+\theta]$.

The details of how we generate each $\mathbf{x}$ are as follows. We use circular random variables in order to ensure that there is no preferential orientation. We generate (x_{21},x_{22}) from (x_{11},x_{12}) in the way described in order to ensure that the marginal PDF's $P(x_{11},x_{12})$ and $P(x_{21},x_{22})$ are the same. We randomly rotate within $[-\theta,+\theta]$ in order to ensure that (x_{11},x_{12}) and (x_{21},x_{22}) are not completely correlated yet not completely independent in a way that is controlled by the size of θ. The limit $\theta=0$ gives $P(\mathbf{x})=P(x_{11},x_{12})\delta(x_{21}-x_{11})\delta(x_{22}-x_{12})$ (i.e. identically correlated), and the limit $\theta=\pi$ gives $P(\mathbf{x})=P(x_{11},x_{12})P(x_{21},x_{22})$ (i.e. completely independent). The overall effect of this prescription for generating inputs $\mathbf{x}$ is to create a training set with fixed marginal PDF's and programmable correlations, as we require in order to demonstrate self-supervision in a carefully controlled way.

2. Compute Nearest Neighbours

We specify the stage 0 codebooks by $\mathbf{x}'_1(y_1)=(x'_{11}(y_1),x'_{12}(y_1))$ and $\mathbf{x}'_2(y_2)=(x'_{21}(y_2),x'_{22}(y_2))$, so the nearest neighbour encoding prescription yields

$$y_1^0(x_{11},x_{12}) = \underset{y_1}{\arg\min}\left((x'_{11}(y_1)-x_{11})^2+(x'_{12}(y_1)-x_{12})^2\right)$$
$$y_2^0(x_{21},x_{22}) = \underset{y_2}{\arg\min}\left((x'_{21}(y_2)-x_{21})^2+(x'_{22}(y_2)-x_{22})^2\right) \quad (2)$$

This is a standard procedure which should need no further explanation. For brevity, denote the result of this operation as $y^0=(y_1^0,y_2^0)$. Encoding mode NN uses y^0, whereas encoding mode MD refines y^0 somewhat in the ensuing steps.

3. Compute Distortion PDF's

Channel mode C: The joint distortion PDF is $P(y'|y)=P(y'_1,y'_2|y_1,y_2)$, whose two marginal PDF's $P(y'_1|y_1,y_2)$ and $P(y'_2|y_1,y_2)$ specify the topographic neighbourhoods of the stage 0 codebooks.

Channel mode I: We also use the two reduced marginal PDF's $P(y'_1|y_1)$ and $P(y'_2|y_2)$ to perform a control simulation in which the pair of VQ's are trained independently. The mode I simulation acts as a control to check that the self-supervision effects that we observe in the mode C simulation genuinely arise from the transfer of information between the pair of VQ's.

We derive $P(y'_1|y_1,y_2)$ (and $P(y'_2|y_1,y_2)$) from $P(y_1,y_2)$ using a heuristic procedure which we may obtain from the following simplification of Equation 35

$$P(y_1'|y_1,y_2) \propto \rho(y'_1,y_2)\exp(-\pi\rho(y_1,y_2)(y_1'-y_1)^2) \tag{3}$$

where we retain only those terms that depend on y'_1. We may interpret the terms in Equation 3 as follows. The exponential factor determines the envelope of values of y'_1-y_1 that are permitted by the PDF, and the $\rho(y'_1,y_2)$ factor provides a bias that weights the PDF in the direction of increasing code vector density. If we recall that $\rho \propto P^{N/(N+2)} \propto P^{1/2}$ (for our N=2 dimensional VQ's), then we may replace the ρ factors in Equation 3 by $P^{1/2}$ factors. In our simulations we shall go one step further by approximating Equation 3 as

$$P(y_1'|y_1,y_2)=\begin{cases}\pi(y_1'-y_1) & G>\kappa \\ (\pi(y_1'-y_1)+\pi(y_1-y_1'))/2 & |G|\le\kappa \\ \pi(y_1-y_1') & G<-\kappa\end{cases} \tag{4}$$

where $G\equiv\partial P(y_1,y_2)/\partial y_1$. We display a typical set of distortion PDFs as histograms plotted against $\Delta y=y'_1-y_1$ in Figure 6

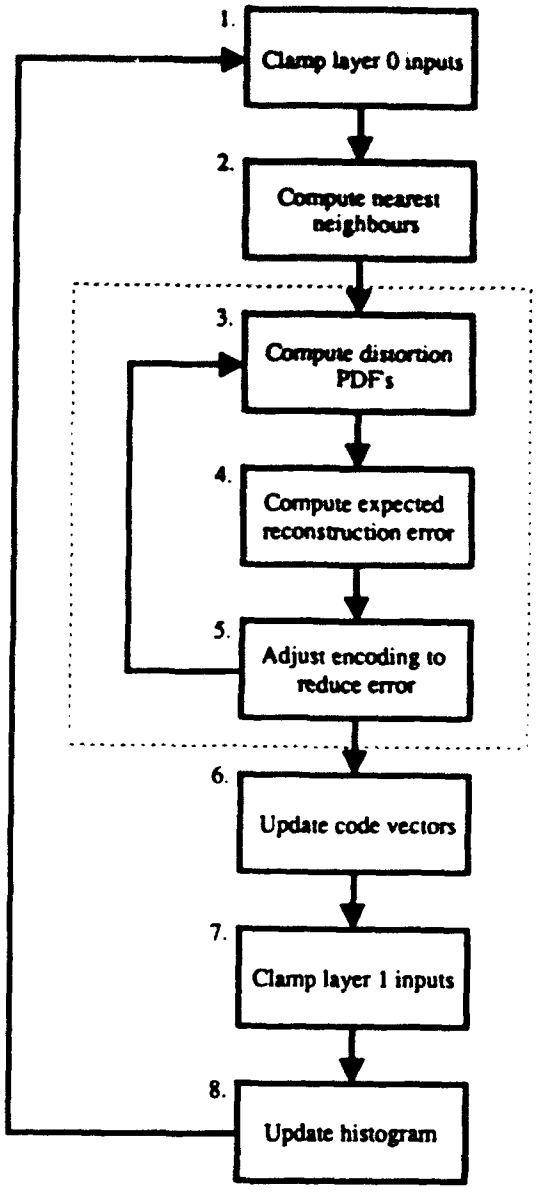


Figure 5. Flowchart showing the main steps in simulating a 2-stage vector quantiser, with the second stage implemented as an ensemble average vector quantiser. The section in the dashed box is an optional minimum distortion encoding scheme, which refines the encoding found by the nearest neighbour scheme.

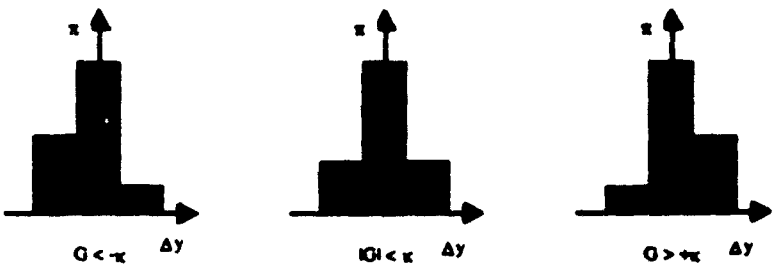


Figure 6. Typical approximations to the distortion PDF $P(y'_1|y_1,y_2)$. We use a very simple prescription in which $P(y'_1|y_1,y_2)$ is set to one of only three possible PDF's according to the value of the underlying gradient $G=\partial P(y_1,y_2)/\partial y_1$. These histograms may be applied directly to the stage 0 VQ's as topographic neighbourhood functions.

The whole of Equation 4 (and Figure 6) is specified by the values of just 3 numbers. Each $\partial P(y_1,y_2)/\partial y_1$ is specified by the values of 3 numbers. We choose to define them as follows

$$\pi(y_1 - y'_1) = \begin{cases} \pi_- & \Delta y = -1 \\ \pi_0 & \Delta y = 0 \\ \pi_+ & \Delta y = +1 \end{cases} \tag{5}$$

	π_-	π_0	π_+
1	0.35	0.60	0.05
2	0.30	0.60	0.10
3	0.25	0.60	0.15
4	0.20	0.60	0.20
5	0.15	0.60	0.25
6	0.10	0.60	0.30
7	0.05	0.60	0.35

Table 2. Values of $\pi_\pm$ and π_0 that we use in 7 separate numerical experiments. Experiment 4 uses an unbiassed distortion, experiments 5-7 use a distortion that is biassed in the direction of increasing PDF (see Figure 6), and experiments 1-3 are biassed in the opposite direction. Only experiments 5-7 have a distortion that corresponds to the one required by theory.

In Table 2 we list the values of π_2 and π_0 that we use in 7 separate numerical simulations. We use a variety of values in order to investigate the effect of both positive and negative biasses, i.e. biasses both in the direction of and opposite to the gradient of $P(y_1,y_2)$. Note that only a positive bias corresponds to the requirements of Equation 3, where $\rho(y'_1,y_2)$ biasses $P(y'_1|y_1,y_2)$ in the direction of positive $\partial P(y_1,y_2)/\partial y_1$.

We may remove the effect of self-supervision by replacing Equation 3 by

$$P(y'_1|y_1) \propto \rho(y'_1)\exp\left(-\pi\rho(y_1)(y'_1-y_1)^2\right) \tag{6}$$

where $\rho(y_1)$ depends on the *marginal* PDF $P(y_1)$. Naturally, we can generate the marginal PDF's during a numerical simulation in which we use joint PDF's.

4. Compute Expected Reconstruction Error

From Equation 1 we may write the expected reconstruction error D(x) for the current input vector x as

$$D(\mathbf{x}) = \int dy'_1\, P(y'_1|y_1,y_2)\left((x'_{11}(y_1)-x_{11})^2+(x'_{12}(y_1)-x_{12})^2\right) \;+\; (1\leftrightarrow 2) \tag{7}$$

where we initialise (y_1,y_2) to (y_1^0,y_2^0) the first time we pass through the minimum distortion loop. We evaluate the integral over y'_1 (and y'_2) somewhat crudely as a sum using the appropriate histograms chosen from Figure 6.

5. Adjust Encoding to Reduce Error

Now that we have calculated the expected reconstruction error D(x) for our *initial* guess (y_1^0,y_2^0) at the correct values of $y_1(x_{12},x_{12})$ and $y_2(x_{21},x_{22})$, we must investigate how it varies in the vicinity of (y_1^0,y_2^0). We may then locate the local minimum (y_1,y_2) of D(x), which in general will *not* be $(y_1,y_2)=(y_1^0,y_2^0)$ (i.e. minimum distortion encoding is not the same as nearest neighbour encoding). Note that for each alternative value of (y_1,y_2) that we investigate we must repeat steps 3 and 4 in order to determine the corresponding value of D(x). In our simulations we explore only the immediate neighbourhood of (y_1^0,y_2^0) given by

$$\begin{aligned} y_1 &\in \{y_1^0, y_1^0 \pm 1\} \\ y_2 &\in \{y_2^0, y_2^0 \pm 1\} \end{aligned} \tag{8}$$

This rather limited search for the minimum distortion encoding succeeds only because we choose to use the distortion PDF's $P(y'_1|y_1,y_2)$ and $P(y'_2|y_1,y_2)$ that we show in Figure 6. If the range of these distortion PDF's were greater, then we would have to consider using a longer range search procedure.

6. Update Code Vectors

From Equation 19b we may write the code vector update prescription as

$$\Delta x'_k(y_k) = \varepsilon P(y_k|y_1,y_2)\left(x_k - x'_k(y_k)\right) \tag{8}$$

where (y_1,y_2) is the minimum distortion encoding located in steps 3-5 above. In our numerical simulations we use ε=0.1 throughout the optimisation; we do not gradually reduce ε to zero.

7. Clamp Layer 1 Inputs

Now that we have finally decided what (y_1,y_2) should be output by the encoders $(y_1(x_1),y_2(x_2))$ we may use this to clamp the inputs to layer 2.

8. Update Histogram

Layer 2 contains a (leaky) histogram representation of $P(y_1,y_2)$ which we now update according to the prescription in Equation 42, with the decay term implemented as $r(k)\rightarrow r(k)/e$ after every $1/(1-\beta)$ time steps. In our simulations we use a memory time of $1/(1-\beta)$=100.

3.2. Splitting procedure

In [4] we presented in detail a phenomenological distortion model that we used to obtain an efficient training procedure for topographic mappings and their application to multistage VQ's. Alternatively, we could use the standard topographic mapping training procedure in [12], but this is a rather inefficient algorithm. It is much more efficient to use a splitting procedure where we perform a crude optimisation using 2 codevectors, which we then use to initialise a more refined optimisation using 4 codevectors, and so on. In our simulations we stop at 8 codevectors. This "coarse to fine" strategy is very effective at rapidly producing an optimum set of codevectors.

In our numerical simulations we optimise each generation of code vectors using 50 training vectors per code vector, before splitting to produce the initial code vector configuration in the next generation.

3.3. Experimental results

We now present the results of several numerical simulations conducted according to procedure that we have described. We run each simulation 4 times to cover the possibilities NN/I, NN/C, MD/I and MD/C that we show in Table 1.

In Figure 7, 8 and 9 we present the NN/I and NN/C results, and in Figure 10, 11 and 12 we present the MD/I and MD/C results. In each Figure we present two plots for channel modes I and C. The dashed lines indicate error bar envelopes, where each point that we plot is the average of the value of D obtained from 16 independent optimisation simulations (in each simulation we accumulate statistics for 256 test set samples to estimate D).

In the case of symmetric distortion (i.e. entry number 4 in Table 2) the I and C plots produce the same value of D. This is because this type of distortion forces $P(y'_1|y_1,y_2)=P(y'_1|y_1)$. This is a simple check of the consistency of our I and C results.

For positively biassed distortions (i.e. in accord with theory) the C plots are systematically lower than the I plots. This behaviour demonstrates convincingly that self-supervision produces a reduced reconstruction error, whether NN or MD encoding is used.

For negatively biassed distortions (i.e. in contradiction with theory) the C plots are systematically higher than the I plots. Because we use an artificially incorrect distortion it is difficult to interpret this result closely. It merely corroborates what we might have expected to happen when we ignore what the theory tells us to do.

When we study the effect of encoding mode, we discover that the MD plots are systematically lower than the corresponding NN plots. This behaviour demonstrates convincingly that a full search for the appropriate encoding is better than a partial search, whether I or C channels are used. This is to be expected.

As the correlation between the channels is reduced (i.e. increase the input correlation angle), the difference between the I and C plots systematically decreases, *except* for Figure 12 where we present the MD/I and MD/C plots for uncorrelated inputs. We would expect that I and C plots should overlap in Figure 12 and in Figure 9, because there are no correlations between the channels. However Figure 12, and to a lesser extent Figure 9, show a clear departure from this expectation. This apparent failure occurs because the histograms suffer from Poisson statistics, so they do not record independent channel statistics (i.e. what is recorded in the histograms does *not* satisfy $P(y_1,y_2)\neq P(y_1)P(y_2)$), so the C simulation is affected by these spurious correlations to produce results that differ from the I simulation.

Taken together Figures 7-12 demonstrate the consistency of our numerical simulations, and demonstrate the benefits of self-supervision (and, coincidentally, minimum distortion encoding) when a simple network is applied to an artificially constructed set of data. These results corroborate the theoretical results that we presented earlier.

4. Conclusions

The main result that we present in this memorandum is the theoretical derivation of and numerical simulation of the phenomenon of *self-supervision*. For illustrative purposes we consider the problem of a pair of communication channels that cause mutual distortion. In our numerical simulations we present a simple demonstration of the improvement in performance that we can obtain be jointly optimising the pair of communication channels, compared with independent optimisation.

In order to make contact with the theory of unsupervised adaptive networks we model the communication channel problem as a nested VQ, as shown in Figure 2. The effect of the inner VQ models the mutual channel distortion, and thus influences the way in which the outer VQ's must be optimised. This is the phenomenon of self-supervision, where one part of an overall unsupervised network supervises the optimisation of another part of the network.

This principle may easily be generalised to a multilayer unsupervised network, although we have not done so in this memorandum. This would mean that we operate an unsupervised multilayer network in such a way that it supervises its own internal operation by passing control signals back from higher layers to lower layers, which in turn causes the lower layers to process their inputs more effectively.

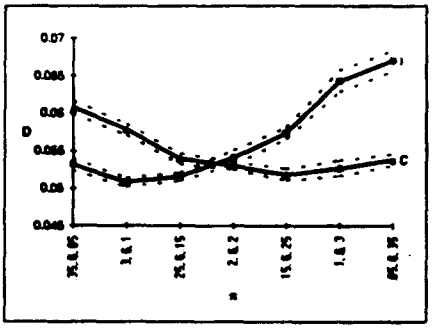


Figure 7. NN, θ=0.5

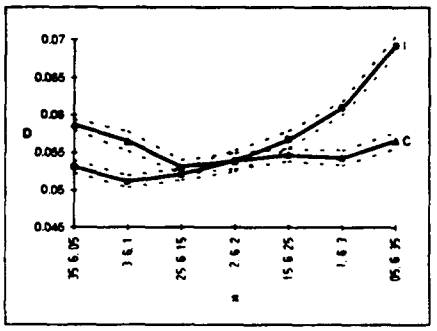


Figure 8. NN, θ=1.0

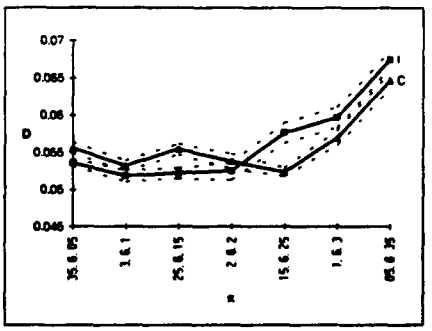


Figure 9. NN, θ=π

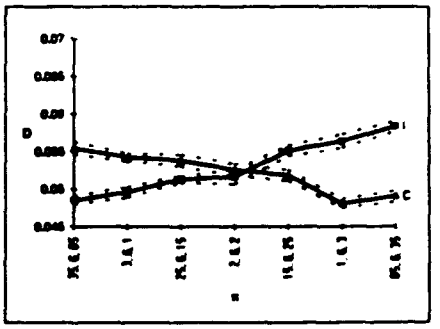


Figure 10. MD, θ=0.5

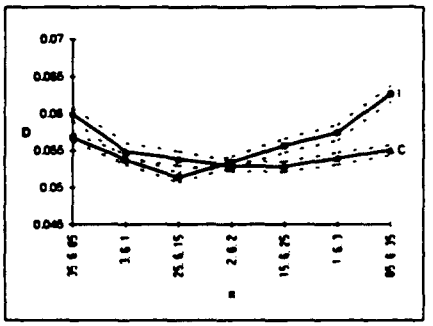


Figure 11. MD, θ=1.0

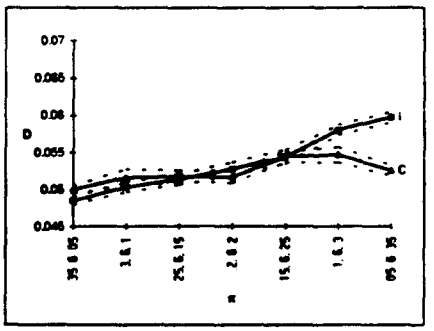


Figure 12. MD, θ=π

5. Recommendation

The results that we present in this memorandum suggests an interesting new direction of research into multistage LVQ networks. The LVQ approach [1] to training a classifier network has had limited success because it uses a single stage VQ (with supervision). Our multistage VQ network could be supervised in the same way as in the LVQ method, and the training signals backpropagated down through the layers of the network. At each layer the backpropagating signals would consist of two components: a term which requires the code vectors to be updated (i.e. a programmable topographic neighbourhood, as in self-supervision), plus a term which requires the layer's inputs to be updated. The first type of term is familiar from our experience with self-supervision, whereas the second term is new (it did not occur in our simulations because we did not experiment with multilayer networks).

6. Notation and terminology

Single stage vector quantiser:

$\mathbf{x}$ = input data

$\mathbf{y}$ = compressed data

$\mathbf{x}'$ = reconstruction of the input data

$\mathbf{y}(\mathbf{x})$ = compression operation, mapping $\mathbf{x} \rightarrow \mathbf{y}$

$\mathbf{x}'(\mathbf{y})$ = reconstruction operation, mapping $\mathbf{y} \rightarrow \mathbf{x}'$

$P(\mathbf{x})$ = PDF of input data

$P(\mathbf{y})$ = PDF of channel data

Two-stage vector quantiser:

$\mathbf{x}_1$ = input data (channel 1)

$\mathbf{x}_2$ = input data (channel 2)

$\mathbf{y}_1$ = compressed data (channel 1)

$\mathbf{y}_2$ = compressed data (channel 2)

$\mathbf{y}'_1$ = distorted compressed data (channel 1)

$\mathbf{y}'_2$ = distorted compressed data (channel 2)

$\mathbf{x}'_1$ = reconstruction of the input data (channel 1)

$\mathbf{x}'_2$ = reconstruction of the input data (channel 2)

$\mathbf{y}_1(\mathbf{x}_1)$ = compression operation (channel 1), mapping $\mathbf{x}_1 \rightarrow \mathbf{y}_1$

$\mathbf{y}_2(\mathbf{x}_2)$ = compression operation (channel 2), mapping $\mathbf{x}_2 \rightarrow \mathbf{y}_2$

$\mathbf{x}'_1(\mathbf{y}'_1)$ = reconstruction operation (channel 1), mapping $\mathbf{y}'_1 \rightarrow \mathbf{x}'_1$

$\mathbf{x}'_2(\mathbf{y}'_2)$ = reconstruction operation (channel 2), mapping $\mathbf{y}'_2 \rightarrow \mathbf{x}'_2$

$\mathbf{z}(\mathbf{y}_1,\mathbf{y}_2)$ = compression operation (fusing channel 1 and channel 2)

$y'_1(z)$ = reconstruction operation (recovering channel 1)

$y'_2(z)$ = reconstruction operation (recovering channel 2)

$P(x_1,x_2)$ = joint PDF of input data (channel 1 and channel 2)

$P(x_1)$ = marginal PDF of input data (channel 1)

$P(x_2)$ = marginal PDF of input data (channel 2)

$P(y_1,y_2)$ = joint PDF of compressed data (channel 1 and channel 2)

$P(y'_1,y'_2|y_1,y_2)$ = conditional PDF of distorted compressed data (channel 1 and channel 2)

$P(y'_1|y_1,y_2)$ = marginal conditional PDF of distorted compressed data (channel 1)

$P(y'_2|y_1,y_2)$ = marginal conditional PDF of distorted compressed data (channel 2)

Note that we use the terms compress/encode (and reconstruct/decode) interchangeably.

We also use the generic notation $P(\cdot)$ to denote a PDF, so unless we state otherwise the functional form of $P(\cdot)$ may be deduced from the nature of the argument that we insert into the function.

Finally, we use the word "stage" to denote a pair of adjacent layers in a multilayer network. Thus "stage 0" means "layers 0 and 1". We use this terminology to refer to the transformation *between* layers, rather than the layers themselves.

7. Appendix

In this section we present a resumé of the VQ theory of Linde et al [10] (the LBG algorithm), and its extension to multistage VQ's [2, 3, 4, 5]. These extensions are related to the VQ theory of Kumazawa et al [11] for communication over a noisy channel, and to the topographic mapping theory of Kohonen [12] for training self-organising neural networks[10].

7.1. Vector quantisation

Define $\mathbf{x}$ as the input data, $\mathbf{y}$ as the compressed data, and $\mathbf{x}'$ as the reconstruction of the input data. Define $\mathbf{y}(\mathbf{x})$ as the compression operation $\mathbf{x} \rightarrow \mathbf{y}$, and $\mathbf{x}'(\mathbf{y})$ as the reconstruction operation $\mathbf{y} \rightarrow \mathbf{x}'$, which yields overall $\mathbf{x}'=\mathbf{x}'(\mathbf{y}(\mathbf{x}))$. Note that the compression and reconstruction process may be interpreted in terms of encoding and decoding during transmission of information through a noiseless communication channel[11], as shown in Figure 1.

We may combine these quantities to obtain the average L_2 (i.e. Euclidean) distortion D_1

---

[10] Hierarchical VQ theory and topographic mapping theory are not exactly equivalent, but the utility of allowing topographic-type mappings to emerge naturally from minimisation of a Lyapunov function cannot be overemphasised. It is tempting to suggest that Kohonen should have formulated his theory in this way in the first place.

[11] Our results could indeed be applied to the optimisation of VQs for communication of information through noisy communication channels, but that *is not the purpose of our research programme*. Our primary motivation for using a VQ model is to obtain simple closed-form analytic solutions, which we may then use to develop our understanding of more complicated models in the future.

$$D_1 = \int dx\, P(x) \| x'(y(x)) - x \|^2 \tag{9}$$

The VQ whose (continuum limit) codebook is defined by the pair of functions $(y(x), x'(y))$ can be optimised by minimising D_1 with respect to variations of $y(x)$ and $x'(y)$.

7.1.1. Vector quantisation for a noisy channel

A more general form of Equation 9 that gives the average L_2 distortion for a VQ with a noisy communication channel [3, 4, 5, 11] is

$$D_2 = \int dx\, P(x) \int dy'\, \pi(y' - y(x)) \| x'(y') - x \|^2 \tag{10}$$

In Equation 10 we assume that $y' = y(x) + n$, where n is a random noise variable with PDF $\pi(n)$, so $P(x,n) = P(x)\pi(n)$, assuming x and n are independent.

7.1.2. Nearest neighbour versus minimum distortion encoding

We functionally differentiate D_2 to calculate the zeros of $\delta D_2/\delta y(x)$ and $\delta D_2/\delta x'(y)$, which yields[12] (see [3, 4, 5] for the details)

$$y(x) = \underset{y}{\arg\min} \int dy'\, \pi(y' - y) \| x'(y') - x \|^2 \tag{11}$$

$$x'(y) = \frac{\int dx\, P(x)\, \pi(y - y(x))\, x}{\int dx\, P(x)\, \pi(y - y(x))} \tag{12a}$$

$$\Delta x'(y) = \varepsilon\, \pi(y - y(x))(x - x'(y)) \tag{12b}$$

In Equation 12 there are two methods of updating $x'(y)$.

1. Batch update (Equation 12a): This is equivalent to one cycle of the LBG algorithm [10].
2. Continuous update (Equation 12b): This is identical to the topographic mapping training algorithm [12], so $\pi(n)$ can be interpreted as a topographic neighbourhood function[13].

In Equation 11 there are two distinct cases to consider.

---

[12] The fact the output of the encoder is usually a discrete variable does not invalidate our use of an assumed continuous output in our derivations. We use continuum derivations because it is simpler to see what is going on. Afterwards, we convert our continuum derivations into discrete derivations by exchanging integrals for sums, derivatives for finite differences, etc. Note that it is not in general easy to convert in the opposite direction (i.e. from a discrete calculation into a "smooth" continuum calculation), but this does not affect our results.

[13] This proves to be a very fertile way of theoretically handling topographic mapping phenomena.

1. Nearest Neighbour Encoding (NN): When $\pi(n)=\delta(n)$ (i.e. the noiseless case) Equation 11 specifies a *nearest neighbour* encoding prescription $y^0(x)$, i.e. given x select as $y=y^0(x)$ the y that minimises $\|x'(y)-x\|^2$.

2. Minimum Distortion Encoding (MD): When $\pi(n)\neq\delta(n)$ Equation 11 specifies a *minimum distortion* encoding prescription $y(x)$, where the effect of $\pi(n)$ is anticipated when selecting $y(x)$.

NN encoding can be used as an approximation to MD encoding when $\pi(n)\neq\delta(n)$. In order to compare NN with MD encoding we develop a Taylor series expansion of the $\|x'(y')-x\|^2$ factor in Equation 11 about its stationary point $y'=y^0(x)$

$$\left\|x'(y')-x\right\|^2 = d^0 + \tfrac{1}{2}\sum_{i,j}(y'-y^0)_i(y'-y^0)_j d^2_{ij} + \tfrac{1}{6}\sum_{i,j,k}(y'-y^0)_i(y'-y^0)_j(y'-y^0)_k d^3_{ijk} + \cdots \quad (13)$$

where d^0, d^2_{ij} and d^3_{ijk} are the zeroth, second and third derivatives of $\|x'(y)-x\|^2$ at $y=y^0(x)$

$$d^0 \equiv \left\|x'(y^0(x))-x\right\|^2$$

$$d^2_{ij} \equiv \frac{\partial^2}{\partial y_i \partial y_j}\left\|x'(y)-x\right\|^2\Big|_{y=y_0(x)} \quad (14)$$

$$d^3_{ijk} \equiv \frac{\partial^3}{\partial y_i \partial y_j \partial y_k}\left\|x'(y)-x\right\|^2\Big|_{y=y_0(x)}$$

whence

$$D_2(x) = \int dy'\, \pi(y'-y)\, \|x'(y')-x\|^2 =$$

$$d^0 + \tfrac{1}{2}\sum_{i,j}\left(\pi^2_{ij}+\pi^1_i(y-y^0)_j+\pi^1_j(y-y^0)_i+(y-y^0)_i(y-y^0)_j\right) d^2_{ij} +$$

$$\tfrac{1}{6}\sum_{i,j,k}\left(\pi^3_{ijk}+\left(\pi^2_{ij}(y-y^0)_k+2\text{ sim.}\right)+\left(\pi^1_i(y-y^0)_j(y-y^0)_k+2\text{ sim.}\right)+(y-y^0)_i(y-y^0)_j(y-y^0)_k\right) d^3_{ijk} + \cdots \quad (15)$$

where we have defined the first three moments of $\pi(n)$ as

$$\pi^1_i \equiv \int dn\, \pi(n)\, n_i$$

$$\pi^2_{ij} \equiv \int dn\, \pi(n)\, n_i n_j \quad (16)$$

$$\pi^3_{ijk} \equiv \int dn\, \pi(n)\, n_i n_j n_k$$

To locate the minimum distortion encoding $y(x)$ we must minimise the expression given by Equation 15 with respect to y.

If we ignore the skewness matrix d^3_{ijk} then we can reduce the problem to minimising $\sum_{ij}(y-y^0+\pi^1)_i(y-y^0+\pi^1)_j d^2_{ij}$. The Hessian matrix d^2_{ij} has no negative eigenvalues, because in Equation 13 we expanded about a local minimum of $\|x'(y)-x\|^2$, so we obtain

$$y(x) = y^0(x) - \pi^1 \qquad (17)$$

The solution is shifted away from the nearest neighbour encoding $y^0(x)$ by an amount equal to *minus* the bias that $\pi(n)$ introduces. This is intuitively reasonable because the effect of minimum distortion encoding anticipates the distorting effect of $\pi(n)$, and will compensate for any bias that $\pi(n)$ introduces.

When $\pi^1=0$ it is important to retain d^3_{ijk} because it is then the lowest order contribution to the difference between $y(x)$ and $y^0(x)$.

7.1.3. Mean field versus local field optimisation

The update prescription depends on the biassed marginals $P(y_1'|y_1,y_2)$ and $P(y_2'|y_1,y_2)$, which causes a migration of $P(y_1,y_2)$ as shown in Figure 13.

The widths of the marginals $P(y_1'|y_1,y_2)$ and $P(y_2'|y_1,y_2)$ determine the widths of the vertical and horizontal bands of $P(y_1,y_2)$ that are affected. It is these changes in $P(y_1,y_2)$ (and hence $P(y_1'|y_1,y_2)$ and $P(y_2'|y_1,y_2)$) that cause the differences between the "mean field" and "local field" optimisation procedures.

Strictly speaking, the change to $P(y_1,y_2)$ is not restricted entirely to the vicinity of the two regions indicated in Figure 13. For instance, the movement of the code vectors in the topographic neighbourhood of $y_1(x)$ and $y_2(x)$ can change the shape of the quantisation cells of other code vectors, which, in turn, causes other changes to $P(y_1,y_2)$. However, this is a second order effect.

We see from Figure 13 that the net migration averaged over all inputs has the affect of *squeezing* the $P(y_1,y_2)$ distribution. This inward pressure is counterbalanced by the *stretching* tendency of each marginal $P(y_1)$ and $P(y_2)$ to become approximately uniform, as normally occurs in VQ's[14].

When we perform a "mean field" simulation we do not take account of these changes to $P(y_1,y_2)$ (and hence $P(y_1'|y_1,y_2)$ and $P(y_2'|y_1,y_2)$) when we calculate the gradient of D in Equation 1. However, in our simulations we represent $P(y_1,y_2)$ as a slowly drifting histogram, so the changes to $P(y_1,y_2)$ gradually become felt later on in the simulation. This does *not* mean that we effectively take $P(y_1,y_2)$ variations into account in a "mean field" simulation, because when the "mean field" simulation reaches equilibrium so that the drift of $P(y_1,y_2)$ vanishes, the "local field" gradient of $P(y_1,y_2)$ does *not* vanish.

---

[14] *Informally, we can interpret this competition as tending to maximise the mutual information* $I[y_1;y_2]$ *between* y_1 *and* y_2. *Thus* $I[y_1;y_2]=H[y_1]+H[y_2]-H[y_1,y_2]\geq 0$, *where* $H[\cdot]$ *is the entropy of its argument, and stretching causes* $H[y_1]$ *and* $H[y_2]$ *to increase, whereas squeezing causes* $H[y_1,y_2]$ *to decrease, hence* $I[y_1;y_2]$ *tends to increase, although this is not absolutely guaranteed. Mutual information can be used as our basic optimisation criterion instead of* L_2 *distortion minimisation. An example of this approach and its relationship to the optimisation of a novel class of hierarchical Gibbs distributions can be found in Luttrell [15,16].*

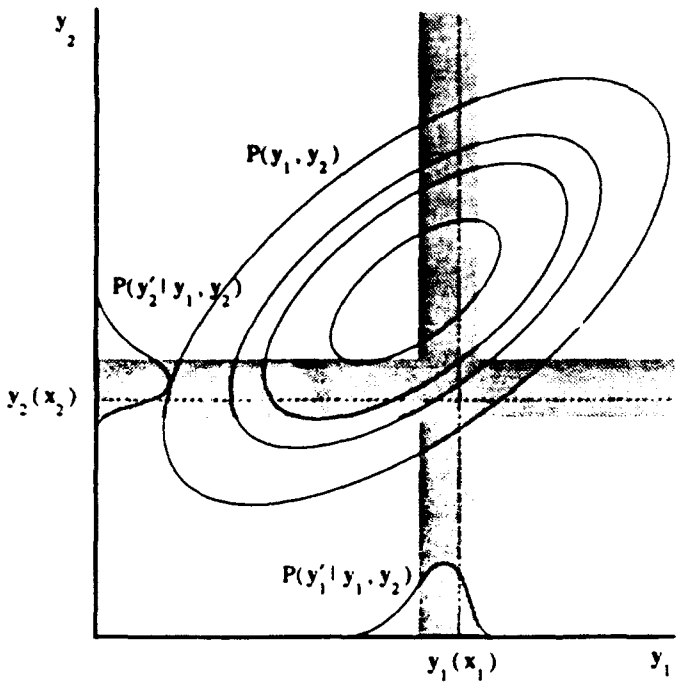


Figure 13. Migration of the PDF $P(y_1,y_2)$ due to the self-supervision effect of the marginal PDFs $P(y'_1|y_1,y_2)$ and $P(y'_2|y_1,y_2)$, which are the topographic neighbourhood functions for optimising the $x_1 \rightarrow y_1 \cdots y_1' \rightarrow x_1'$ and $x_2 \rightarrow y_2 \cdots y_2' \rightarrow x_2'$ channels. Contributions to $P(y_1,y_2)$ which lie inside the vertical shaded band tend to migrate towrds the left, and contributions inside the horizontal band tend to move upwards. In all cases the migration is in the direction in which the corresponding marginal PDF is biased. Compare Figure 4.

7.1.4. Two-stage vector quantising

When we minimise D using the mean field procedure we obtain

$$(y_1(x_1),y_2(x_2)) = \underset{(y_1,y_2)}{\arg\min} \left(\int dy'_1 P(y'_1|y_1,y_2) \, \|x'_1(y'_1)-x_1\|^2 + (1 \leftrightarrow 2) \right) \tag{18}$$

$$x'_k(y_k) = \frac{\int dx_1 \, dx_2 \, P(x_1,x_2) \, P(y_k|y_1(x_1),y_2(x_2)) \, x_k}{\int dx_1 \, dx_2 \, P(x_1,x_2) \, P(y_k|y_1(x_1),y_2(x_2))} \tag{19a}$$

$$\Delta x'_k(y_k) = \varepsilon \, P(y_k|y_1(x_1),y_2(x_2)) \, (x_k-x'_k(y_k)) \tag{19b}$$

which should be compared with Equation 11 and Equation 12. Note that Equation 18 specifies a minimum distortion prescription in which we *simultaneously* optimise $y_1(x_1)$

and $y_2(x_2)$, using $P(y'_1|y_1,y_2)$ and $P(y'_2|y_1,y_2)$ instead of $\pi(y'-y)$. We sometimes approximate this by using a nearest neighbour prescription. Note that in [13] we reported (in the case of scalar quantisation) that this approximation is forbidden if we wish the density of code vectors to be insensitive to the choice of $P(y'_2|y_1,y_2)$ and $P(y'_1|y_1,y_2)$.

Note that in [13] we reported (in the case of scalar quantisation) that the use of nearest neighbour encoding is forbidden if we wish the density of code vectors to be insensitive to the choice of $P(y'_2|y_1,y_2)$ and $P(y'_1|y_1,y_2)$.

7.2. Analytically solvable quantisation model

In this section we present an analytically solvable model of the ensemble distortion $P(y'_1,y'_2|y_1,y_2)$ shown in Figure 4.

7.2.1. Code vector density

If the number of code vectors $(y'_1(z),y'_2(z))$ in the $(y_1,y_2) \rightarrow z \rightarrow (y'_1,y'_2)$ codebook is very large, then we may calculate $P(y'_1,y'_2|y_1,y_2)$ directly. Thus, we model the ensemble properties of the codebook by defining $\rho(y_1,y_2)$, which specifies the density of code vectors $(y'_1(z),y'_2(z))$ in (y_1,y_2)-space.

7.2.2. Transition probability: integral equation

Note that we use the notation $\rho(y)$ (and $P(y'|y)$) and $\rho(y_1,y_2)$ (and $P(y'_1,y'_2|y_1,y_2)$) interchangeably..

In Figure 14 we compare the nearest neighbour encoding prescription for a *single* VQ with that for an *ensemble* of VQ's[15].

In Figure 14a we show an input vector (represented by a cross) and the known positions of the code vectors of a single VQ. The nearest neighbour can be located by expanding a circle centred on the input vector until it grazes the nearest code vector, as shown. In Figure 14b we show the ensemble version of the same diagram, in which the precise code vector positions are unknown, so there is a distribution $P(y'|y)$ of possible nearest neighbour locations. There is an analogous interpretation for the minimum distortion encoding prescription.

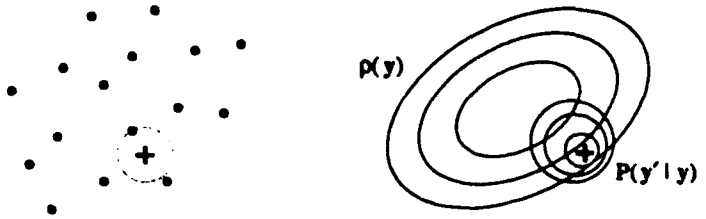
Using Figure 14, we may write down an integral equation that relates $P(y'|y)$ to $\rho(y)$. Thus

$$P(y'|y)\,\delta y' = \left(1 - \int_{\|\xi-y\| \le \|y'-y\|} d\xi\, P(\xi|y)\right) \rho(y')\,\delta y' \qquad (20)$$

---

[15]For completeness, we discuss here the intermediate case where the positions of the code vectors are partially known. The most important way of acquiring partial knowledge is to note the positions of the nearest neighbour code vectors during training. However, such knowledge must be continuously updated because migration of the code vector positions gradually erases any memory of their earlier positions. Partial knowledge lies between the extremes of Figure 5a and Figure 5b, and its analysis is very complicated. We choose to analyse the extreme case in Figure 5b because it underestimates, rather than overestimates, the knowledge about the codevectors that is available.

where the first term on the right hand side is the probability that there is no nearest neighbour code vector in the sphere of radius $\|y'-y\|$ centred on y, and the second term is the probability of finding a code vector in the volume $\delta y'$ at y'. The product of these two terms gives the probability of finding the nearest neighbour code vector in the volume $\delta y'$ at y'.



(a) (b)

Figure 14. (a) Determining the nearest neighbour code vector position for a single vector quantiser. (b) Determining the PDF $P(y'|y)$ of the nearest neighbour code vector position from the code vector density $\rho(y)$ of an ensemble of vector quantisers.

7.2.3. Transition probability: constant code vector density case

We now solve Equation 20 for the case $\rho(y)=\rho_0$=constant. The nearest neighbour code vector is then equally likely to lie in any direction from y, so $P(y'|y)$ must be a function only of the radial distance $\|y'-y\|$, which gives

$$P(\|y'-y\|) = \left(1 - \int_{\|\xi-y\|\le\|y'-y\|} d\xi\, P(\|\xi-y\|)\right)\rho_0 \tag{21}$$

where $\|y'-y\|^2 \equiv (y'-y)^T\cdot(y'-y)$. The integrand is spherically symmetric so we may use the transformation

$$\int_{\|\xi-y\|\le\|y'-y\|} d\xi\, P(\|\xi-y\|) = \alpha_N \int_{\|\xi\|\le\|y'-y\|} d\|\xi\|\, \|\xi\|^{N-1}\, P(\|\xi\|) \tag{22}$$

where α_N is a constant deriving from the angular integration in N dimensions. Differentiate Equation 21 with respect to the upper limit $\|y'-y\|$ of the $\|\xi\|$ integration to yield

$$\frac{dP(\|y'-y\|)}{d\|y'-y\|} = -\alpha_N \rho_0 \|y'-y\|^{N-1} P(\|y'-y\|) \tag{23}$$

and integrate to yield finally

$$P(\|y'-y\|) = P_0 \exp\left(-\frac{\alpha_N \rho_0 \|y'-y\|^N}{N}\right) \tag{24}$$

where P_0 should be adjusted to ensure that $P(\|y'-y\|)$ is normalised correctly. The N=2 case reduces to a Gaussian distribution with $P_0=1/(4\pi^2\rho_0)$.

7.2.4. Transition probability: variable code vector density case

We now extend the previous results to the case

$$\rho(y') = \rho(y) + (y'-y)^T.\nabla\rho(y) \tag{25}$$

which is a first order Taylor expansion of $\rho(y')$ about the point $y'=y$. We anticipate that the first order expansion of $P(y'|y)$ has the form of Equation 24 with an extra factor to account for the angular dependence in Equation 25

$$P(y'|y) = P_0(y)\,(1 + (y'-y)^T.a(y))\,\exp\left(-\frac{\alpha_N \rho(y)\,\|y'-y\|^N}{N}\right) \tag{26}$$

where $a(y)$ has to be determined. Differentiating Equation 20 with respect to y' leads to

$$\frac{\partial P(y'|y)}{\partial y'} = -\left(\frac{\partial}{\partial y'}\int_{\|\xi-y\|\le\|y'-y\|} d\xi\, P(\xi|y)\right)\rho(y') + \frac{P(y'|y)}{\rho(y')}\frac{\partial\rho(y')}{\partial y'} \tag{27}$$

where we used Equation 20 to replace a term by a $P(y'|y)/\rho(y')$ factor. We may now insert the expressions for $\rho(y')$ (Equation 25) and $P(y'|y)$ (Equation 26) into Equation 27, and make use of the results

$$\begin{aligned}\frac{\partial P(y'|y)}{\partial y'} &= \frac{\partial}{\partial y'}\left(P_0(y)\,(1 + (y'-y)^T.a(y))\,\exp\left(-\frac{\alpha_N \rho(y)\,\|y'-y\|^N}{N}\right)\right)\\ &= P_0(y)\,\Big(a(y) - \alpha_N \rho(y)\,\|y'-y\|^{N-2}(y'-y)(1+(y'-y)^T.a(y))\Big)\cdot\\ &\quad \exp\left(-\frac{\alpha_N \rho(y)\,\|y'-y\|^N}{N}\right)\end{aligned} \tag{28}$$

which we obtain by using $\partial/\partial y'\|y'-y\|^N = N\|y'-y\|^{N-2}(y'-y)$,

$$\begin{aligned}&\frac{\partial}{\partial y'}\int_{\|\xi\|\le\|y'-y\|} d\xi\, P(\xi|y)\\ &= \frac{y'-y}{\|y'-y\|}\int_{\|\xi\|=\|y'-y\|} dS_\xi\, P(\xi|y)\\ &= P_0(y)\alpha_N\|y'-y\|^{N-2}(y'-y)\exp\left(-\frac{\alpha_N \rho(y)\,\|y'-y\|^N}{N}\right)\end{aligned} \tag{29}$$

which we obtain by using Equation 22 to perform the angular integration over the surface $\|\xi\|=\|y'-y\|$, and noting that the term containing $(y'-y)^T.a(y)$ vanishes after angular integration,

$$\frac{\partial\rho(y')}{\partial y'} = \nabla\rho(y') \tag{30}$$

to obtain

$$\begin{aligned}&a(y) - \alpha_N \rho(y)\,\|y'-y\|^{N-2}(y'-y)(1+(y'-y)^T.a(y)) =\\ &-\alpha_N \rho(y')\,\|y'-y\|^{N-2}(y'-y) + (1+(y'-y)^T.a(y))\frac{\nabla\rho(y')}{\rho(y')}\end{aligned} \tag{31}$$

We then use $\rho(y')\approx\rho(y)(1+(y'-y)^T.\nabla\rho(y)/\rho(y))$ and $\nabla\rho(y')/\rho(y')\approx\nabla\rho(y)/\rho(y)$ to simplify Equation 31 into the form

$$\left(\mathbf{a}(y) - \frac{\nabla\rho(y)}{\rho(y)}\right) \approx \alpha_N \rho(y) \, \|y'-y\|^{N-2} (y'-y) \, (y'-y)^T . \left(\mathbf{a}(y) - \frac{\nabla\rho(y)}{\rho(y)}\right) \tag{32}$$

where we have dropped the next-to-leading order terms. We may solve Equation 32 by choosing $\mathbf{a}(y)=\nabla\rho(y)/\rho(y)$, to obtain the generalisation of Equation 24 as

$$P(y'|y) \approx P_0(y) \left(1 + \frac{(y'-y)^T.\nabla\rho(y)}{\rho(y)}\right) \exp\left(- \frac{\alpha_N \rho(y) \, \|y'-y\|^N}{N}\right) \tag{33}$$

Strictly speaking, Equation 33 does not specify a valid probability distribution because it yields a negative probability when $(y'-y)^T.\nabla\rho(y)/\rho(y) < -1$. However, this result is the leading order term in a Taylor expansion about $y'=y$ (see Equation 25), therefore we implicitly assume $\|(y'-y)^T.\nabla\rho(y)/\rho(y)\| \ll 1$. The effect of the $1+(y'-y)^T.\nabla\rho(y)/\rho(y)$ term is to relocate the maximum of $P(y'|y)$ from its original position at $y'=y$ (see Equation 24) to a new position given by

$$y' \approx y + \sqrt[N-1]{\frac{\|\nabla\rho(y)\|}{\alpha_N \rho(y)^2 \, \|\nabla\rho(y)\|}} \frac{\nabla\rho(y)}{\|\nabla\rho(y)\|} \tag{34}$$

The direction of shift is consistent with the bias in $P(y'|y)$ that we show in Figure 4 and Figure 13.

Finally, we marginalise the joint distribution $P(y_1',y_2'|y_1,y_2)$ in Equation 33 in order to calculate $P(y_1'|y_1,y_2)$ and $P(y_2'|y_1,y_2)$ (which we need in Equation 18 and Equation 19). For the 2-dimensional case ($N=2$, $\alpha_2=2\pi$, $(y_1,y_2)\rightarrow(y_1,y_2)$) this is easy because the exponential factors are Gaussians, leading to the result

$$P(y_1'|y_1,y_2) \approx P_0(y_1,y_2) \left(1 + \frac{(y_1'-y_1)}{\rho(y_1,y_2)} \frac{\partial\rho(y_1,y_2)}{\partial y_1}\right) \exp(-\pi \, \rho(y_1,y_2) \, (y_1'-y_1)^2) \tag{35}$$

with an analogous result for $P(y_2'|y_1,y_2)$. These results may be used to model the marginals in Figure 4.

Recall that $P(y_1'|y_1,y_2)$ and $P(y_2'|y_1,y_2)$ serve as topographic neighbourhood functions for optimising the $\mathbf{x}_1 \rightarrow \mathbf{y}_1 \cdots \mathbf{y}_1' \rightarrow \mathbf{x}_1'$ and $\mathbf{x}_2 \rightarrow \mathbf{y}_2 \cdots \mathbf{y}_2' \rightarrow \mathbf{x}_2'$ transformations. In the ensemble average model, these neighbourhood functions emerge naturally from the ensemble properties of $(y_1,y_2) \rightarrow z \rightarrow (y'_1,y'_2)$, so we do not need to supply them manually.

The automatic generation by one part of a network of the topographic neighbourhood function required by another part of the same network is sufficiently novel and important that we call it *self-supervision*. It is an effect that lies halfway between full supervised training with an external teacher, and unsupervised training. It is an economical way of extending the capabilities of an unsupervised network towards those of a supervised network[16].

[16] Self-supervised networks are unable to produce the outputs required by an external teacher (because there is no teacher). All they can do is to supervise their internal operation, but not that of their output layer.

7.3. Estimating the code vector density

In order to determine the result for $P(y'|y)$ in Equation 33 we must estimate $\rho(y)$ and $\nabla\rho(y)$ in Equation 33. We may use the asymptotic relationship[17]

$$\rho(y) \propto P(y)^{N/(N+2)} \tag{36}$$

or

$$\frac{\nabla\rho(y)}{\rho(y)} = \frac{N}{N+2}\frac{\nabla P(y)}{P(y)} \tag{37}$$

to express our results either in terms of $P(y)$ or $\rho(y)$.

If we record $P(y)$ as a histogram of frequencies of occurrence of input vectors $\mathbf{y}$, then we may directly estimate $\nabla P(y)/P(y)$, and thence estimate $\nabla\rho(y)/\rho(y)$[18]. We may also make a crude estimate of $\nabla\rho(y)/\rho(y)$ from a *single* realisation of the code vectors.

7.3.1. Estimation from the histograms

We will now describe how to estimate $P(y)$ and $\nabla P(y)$ from a histogram. Denote the transformation from continuous to discrete variables as

$$k_i(y_i) \equiv \mathrm{int}\left(\frac{y_i - y_{i,min}}{y_{i,max} - y_{i,min} + \delta} B\right) \tag{38}$$

where $y_{i,min}$ and $y_{i,max}$ are the minimum and maximum values that y_i can possibly take, B is the number of bins in each dimension of the multidimensional histogram, and δ is a small positive number that we introduce to ensure that $0 \le k_i(y_i) < B$ (i.e. a strict inequality at the upper end of the range). Thus the full vector index required to locate a bin in the multidimensional histogram

$$\mathbf{k}(y) = (k_1(y_1), k_2(y_2), \ldots, k_N(y_N)) \tag{39}$$

which we then update using

$$r(\mathbf{k}(y)) \rightarrow r(\mathbf{k}(y)) + 1 \tag{40}$$

It is important that the histogram should also have a finite memory time in order that it can track a time dependent $P(y)$. This is easily arranged by making the histogram bins leaky. For instance, the number of counts in each bin could be a real number (not an integer), all of which simultaneously decay (before $r(\mathbf{k}(y))$ is updated) according to the prescription

$$r(\mathbf{k}) \rightarrow \beta\, r(\mathbf{k}) \tag{41}$$

where $0 < \beta < 1$. The overall update process would then be described by

$$r(\mathbf{k};t+1) = \beta\, r(\mathbf{k};t) + v(\mathbf{k};t) \tag{42}$$

---

[17] Strictly speaking, this is true only for minimum distortion encoding.

[18] We could also substitute for "histogram" any other method of estimating a density, such as a "mixture distribution".

where $\nu(\mathbf{k};t)$ is a multivariate Poisson process which derives from the hits on the array of histogram bins. The "memory time" of this type of process is $1/(1-\beta)$ time steps, so β can readily be used to control the ability of the histogram to track a time dependent $P(\mathbf{y})$.

A less computationally intensive prescription for histogram decay would be to use Equation 42 to decay the histogram bins only occasionally. For instance, if we decay $r(\mathbf{k}) \rightarrow r(\mathbf{k})/e$ after every $1/(1-\beta)$ time steps, then we crudely emulate the effect of Equation 42 applied at every single time step[19]. This leads to quite acceptable results, and it is the procedure that we adopt in our numerical simulations.

7.3.2. Estimation from the code vector positions

For completeness, we shall now describe how to make a crude estimate of $\rho(\mathbf{y})$ and $\nabla\rho(\mathbf{y})$ from a *single* realisation of the code vectors, although we do not make use of this prescription in our numerical simulations. We may obtain the required estimate by measuring the zeroth and first moments ($M_0(\mathbf{y},R)$ and $M_1(\mathbf{y},R)$ respectively) of the code vector positions within a sphere $\|\xi-\mathbf{y}\| < R$. The *definition* of these moments is

$$M_0(\mathbf{y},R) \equiv \int_{\|\xi-\mathbf{y}\|\leq R} d\xi\, \rho(\xi) \qquad M_1(\mathbf{y},R) \equiv \int_{\|\xi-\mathbf{y}\|\leq R} d\xi\, \rho(\xi)\,(\xi-\mathbf{y}) \tag{43}$$

whereas the *estimate* of these moments is

$$M_0(\mathbf{y},R) \equiv \sum_{\|\xi-\mathbf{y}\|\leq R} 1 \qquad M_1(\mathbf{y},R) \equiv \sum_{\|\xi-\mathbf{y}\|\leq R} (\xi-\mathbf{y}) \tag{44}$$

Combining Equation 43 and Equation 44, and inserting the expression in Equation 25 yields

$$\rho(\mathbf{y}) \approx \frac{N}{\alpha_N R^N} \sum_{\|\xi-\mathbf{y}\|\leq R} 1 \qquad \nabla\rho(\mathbf{y}) \approx \frac{N(N+2)}{\alpha_N R^{N+2}} \sum_{\|\xi-\mathbf{y}\|\leq R} (\xi-\mathbf{y}) \tag{45}$$

whence $\nabla\rho(\mathbf{y})/\rho(\mathbf{y})$ is given by

$$\frac{\nabla\rho(\mathbf{y})}{\rho(\mathbf{y})} = \frac{N+2}{R^N} \frac{\sum_{\|\xi-\mathbf{y}\|\leq R} (\xi-\mathbf{y})}{\sum_{\|\xi-\mathbf{y}\|\leq R} 1} = \frac{(N+2)\langle \xi-\mathbf{y} \rangle_{\|\xi-\mathbf{y}\|\leq R}}{R^N} \tag{46}$$

where $\langle \cdots \rangle$ denotes an average over code vectors.

The optimum choice of R is a tradeoff. If R is too small then there are too few code vectors in the sphere $\|\xi-\mathbf{y}\| < R$ to allow a good estimate to be made in Equation 44. If R is too big then we invalidate the assumption that we may ignore the higher order terms (e.g. curvature) in the Taylor expansion in Equation 25. Between these two extremes will lie an optimum choice of R, whose value can be determined by experiment.

---

[19] This can easily be checked as follows $\beta^{1/(1-\beta)} = \exp[\log(\beta)/(1-\beta)] = \exp[\log(1-(1-\beta))/(1-\beta)] = \exp[-(1-\beta-\cdots)/(1-\beta)] = \exp(-1) = 1/e$.

THIS PAGE IS INTENTIONALLY LEFT BLANK

8. References

[1] Kohonen T, Learning vector quantisation, Helsinki University of Technology, 1986, Technical Report TKK-F-A601

[2] Luttrell S P, Self-organising multilayer topographic mappings, Proc. 2nd IEEE Int. Conf. on Neural Networks, San Diego, 1988, **1**, 93-100

[3] Luttrell S P, Self-organisation: a derivation from first principles of a class of training algorithms, Proc. 3rd IEEE Int. Joint Conf. on Neural Networks, Washington DC, 1989, **2**, 495-498

[4] Luttrell S P, Hierarchical self-organising networks, Proc. 1st IEE Conf. on Artificial Neural Networks, London, 1989, 2-6

[5] Luttrell S P, Derivation of a class of training algorithms, IEEE Trans. NN, 1990, **1**, 229-232

[6] Luttrell S P, Hierarchical vector quantisation, Proc. IEE Part I, 1989, **136**, 405-413

[7] Luttrell S P, Image compression using a multilayer neural network, Patt. Recog. Lett., 1989, **10**, 1-7

[8] Luttrell S P, A trainable texture anomaly detector using the Adaptive Cluster Expansion (ACE) method, RSRE, 1990, Technical Report 4437

[9] Luttrell S P, 1991, submitted to IEEE Trans. PAMI, Adaptive Cluster Expansion (ACE): a hierarchical maximum entropy method of estimating probability density functions

[10] Linde Y, Buzo A and Gray R M, An algorithm for vector quantiser design, IEEE Trans. COM, 1980, **28**, 84-95

[11] Kumazawa H, Kasahara M and Namekawa T, A construction of vector quantisers for noisy channels, Electronics and Engineering in Japan, 1984, **67B**, 39-47

[12] Kohonen T, Self organisation and associative memory, Springer-Verlag, 1984

[13] Luttrell S P, Code vector density in topographic mappings: scalar case, IEEE Trans. NN, 1991, **2**, 427-436

[14] Luttrell S P, Asymptotic code vector density in topographic vector quantisers, RSRE, 1990, Technical Report 4392

[15] Luttrell S P, Bayesian inference on a tree, RSRE , 1990, Technical Report SP4/109

[16] Luttrell S P, A hierarchical network for clutter and texture modelling, SPIE Conf. on Adaptive Signal Processing, San Diego, 1991, to appear

THIS PAGE IS INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

DRIC Reference Number (If known) .....

Overall security classification of sheet .....UNCLASSIFIED.....

(As far as possible this sheet should contain only unclassified information. If it is necessary to enter classified information, the field concerned must be marked to indicate the classification eg (R), (C) or (S).

Originators Reference/Report No.	Month	Year
MEMO 4467	DECEMBER	1991

Originators Name and Location
RSRE, St Andrews Road
Malvern, Worcs WR14 3PS

Monitoring Agency Name and Location

Title
SELF-SUPERVISION IN MULTILAYER ADAPTIVE NETWORKS

Report Security Classification	Title Classification (U, R, C or S)
UNCLASSIFIED	U

Foreign Language Title (In the case of translations)

Conference Details

Agency Reference	Contract Number and Period

Project Number	Other References

Authors	Pagination and Ref
LUTTRELL, S P	26

Abstract

We theoretically derive and numerically simulate a new phenomenon called *self-supervision*, in which the higher layers of a multilayer *unsupervised* network control the optimisation of the lower layers, even when there is no external supervising teacher present. Self-supervision is a very convenient hybrid, which combines the best properties of unsupervised and supervised network training algorithms.

Abstract Classification (U,R,C or S)
U

Descriptors

Distribution Statement (Enter any limitations on the distribution of the document)
UNLIMITED

ssssa

THIS PAGE IS INTENTIONALLY LEFT BLANK